Rust And Research

Robert O'Callahan
WHAT IF I TOLD YOU...
A programming language had

Memory safety
A programming language had

Memory safety

No GC
A programming language had

Memory safety
No GC
Stateful updates
A programming language had

Memory safety
No GC
Stateful updates
\textbf{As fast as C/C++}
A programming language had

Memory safety
No GC
Stateful updates
As fast as C/C++
**Integer overflow is an error**
A programming language had

Memory safety
No GC
Stateful updates
As fast as C/C++
Integer overflow is an error
**Data race free**
A programming language had

- Memory safety
- No GC
- Stateful updates
- As fast as C/C++
- Integer overflow is an error
- Data race free

**Tight restrictions on aliasing**
A programming language had

Memory safety
No GC
Stateful updates
As fast as C/C++
Integer overflow is an error
Data race free
Tight restrictions on aliasing
**Affine types (e.g. supports “session types”)**
Sounds like a crazy research project!

Would normal developers be interested in using such a language?

Could it possibly scale to large systems in practice?
Top 15 languages by Github PRs

JavaScript: 1736476
Python: 804790
Java: 703649
Ruby: 560430
PHP: 359040
C++: 319324
TypeScript: 311229
Go: 258131
C#: 246513
CSS: 236795
Shell: 168301
C: 160889
Swift: 67664
Scala: 67188
Rust: 52936
Amazon - Building tools in Rust.

Atlassian (makers of Jira) - Using Rust in the backend.

Dropbox - Using Rust in both the frontend and backend.

Facebook - Tools for source control.

Google - As part of the Fuchsia project.

Microsoft - Using Rust in part of their new Azure IoT work.

npm - Using Rust in some of the npm core services.

Red Hat - Creating a new storage system

Reddit - Using Rust in its comment processing

Twitter - As part of the build team support for Twitter.
Some Rust core principles

Ownership and move semantics

```rust
let x: T = T::new();
let y = x;
```

Borrowed references with lifetimes

```rust
fn f<\'a>(x: &\'a T) → &\'a U { &x.field }
```

Read-only references can be **shared** and the data is **immutable**

Mutable references are **exclusive**

No other reference to that data is in scope
Research problems solved

Memory safety without GC  ✔

Data race freedom  ✔

Practical affine types  ✔

Working in practice at scale!!!  ✔✔✔
New research problems

What sort of static analyses benefit Rust?

Null pointer deref → Option::unwrap()
New research problems

*Can static analyses leverage Rust invariants?*

E.g. mutable references can't alias other references
New research problems

Formal semantics and verification of unsafe Rust

Rustbelt project (Derek Dreyer et al., MPI-SWS)
Conclusions

Rust has raised the bar for systems programming languages

Expect Rust and Rust-like languages to be increasingly used for systems/embedded/safety critical systems

Consider targeting problems relevant to these languages and taking advantage of their features/restrictions